Correction Java Basics - 0

C -

```
import orders.Order;
import orders.items.Item;
```

D -

```
import orders.*;
import orders.items.*;
```

If you check the directory structure, you will find that directory "orders" contains "items", but orders and orders.items are different packages.
import orders.*; will only import all the classes in orders package but not in orders.items package.

Correction Java Basics - 1

F - Welcome Keller!

Class Guest has special main method but main method defined in Message class is not public and hence it can't be called by JVM. But there is no issue with the syntax hence no compilation error.

java Guest Clare Waight Keller passes new String [] {"Clare", "Waight", "Keller"} to args of Guest.main method.

Correction Java Basics - 2

D - public static void main(String [] a)

E - static public void main(String [] args)

As System.out.println needs to be executed on executing the Test class, this means special main method should replace /*INSERT*/.

Special main method's name should be "main" (all characters in lower case), should be static, should have public access specifier and it accepts argument of String [] type. String [] argument can use any identifier name, even though in most of the cases you will see "args" is used.

Correction Java Basics - 3

C -

```
package com.training.oca;

public class Printer {

}
```

If package is used then it should be the first statement, but javadoc and developer comments are not considered as java statements so a class can have developer and javadoc comments before the package statement.

Correction Java Basics - 4

A - Line 10 causes compilation error

E - Line 11 causes compilation error

F - Line 9 causes compilation error

class A is declared public and defined in com.training.oca package, there is no problem in accessing class A outside com.training.oca package.

class TestA is defined in com.training.oca.test package, to use class A either use import statement “import com.training.oca.A;” or fully qualified name of the class com.training.oca.A. No issues at Line 3 and LIne 7.

As TestA is in different package so it can only access public members of class A using object reference. Line 8 compiles successfully.

protected, default and private members are not accessible outside com.training.oca package using object reference.

NOTE: protected members can be accessed outside but only through inheritance and not object reference.

Correction Java Basics - 5

C - Welcome James!

Both the classes contain special main method. No compilation error with the code: file is correctly names as Guest.java (name of public class).

java Guest James Gosling passes new String [] {“James”, “Gosling”} to args of Guest.main method. Apart from being special main method, Message.main is static method so Guest.main method invokes Message.main method with the same argument: new String [] {“James”, “Gosling”}.

args[0] is “James” hence “Welcome James!” gets printed on to the console.

Correction Java Basics - 6

B -

```
package com.training.oca;

import java.util.*;

public class Printer {

}
```

If package is used then it should be the first statement, but javadoc and developer comments are not considered as java statements so a class can have developer and javadoc comments before the package statement.

If import and package both are available, then correct order is package, import, class declaration.

Multiple package statements are not allowed.

Correction Java Basics - 7

C - Compilation Error

Top level class can have two access modifiers: public and default.

Over here Test class has private modifier and hence compilation error.

Correction Java Basics - 8

B - javac MyClass.java

Command to compile a java file: javac .java [.java extension is compulsory]

Command to execute a java class: java [.class extension should not be used]

Correction Java Basics - 9

`D – javac –d classes\ src\com\training\test\Exam.java`

Use -d option with javac command. As you are typing javac command from within Sec07 directory, hence path of java file relative to Sec07 directory needs to be given.

So, correct command is: `javac –d classes\ src\com\training\test\Exam.java`

Correction Java Basics - 10

B - `import com.training.*;`

Following import statements are correct:

```
import com.training.*;
```

```
import com.training.Animal;
```

NOTE: all small case letters in import keyword.

Correction Java Basics - 11

A - Yes

In java, it is allowed to put multiple statements on one line. E.g. below code is legal:

```
public class Test {
    public static void main(String [] args) {
        String symbol = "!";System.out.print("Hello
        ");System.out.print("World");System.out.println(symbol);
    }
}
```

Above code is similar to:

```
public class Test {
    public static void main(String [] args) {
        String symbol = "!";
        System.out.print("Hello ");
        System.out.print("World");
        System.out.println(symbol);
    }
}
```

Empty statements (just the semicolon) are also allowed in java, therefore below code is also legal:

```
public class Test {
    public static void main(String [] args) {
        System.out.println("Hello");
        ;
        ;
        ;
        ;
        ;
        ;
        ;
        ;
    }
}
```

As shown above, java statements (including empty statements) can be placed on one line, therefore below code is legal:

```
public class Test {
    public static void main(String [] args) {
        System.out.println("Hello");;;;;;;;;
    }
}
```

Correction Java Basics - 12

A - HelloWorld.java

Java is case sensitive language. File name should match with public class's name, which is "HelloWorld".

"helloworld" is different from "HelloWorld".

Correction Java Basics - 13

D -

```
package com.exam.oca;
```

To declare Test class in com.exam.oca package, use following declaration:

package com.exam.oca;

No wildcard (*) allowed in package declaration. Don’t include class name in package declaration.

NOTE: all small case letters in package keyword.

Correction Java Basics - 14

A - To print C on to the console, execute below commands:

```
javac Test.java
java C
```

Test.java is a valid java file. As none of the classes in Test.java file are public, hence file name can use any valid Java identifier.

As file name is Test.java, hence to compile the code below command is used:

javac Test.java

Execution of above command creates 4 class files: A.class, B.class, C.class & D.class.

To print C on to the console, class C must be executed. To execute C class, command is:

java C

Correction Java Basics - 15

D - public static void main(String [] a)

F - static public void main(String [] args)

G - public static void main(String… a)

As System.out.println needs to be executed on executing the Test class, this means special main method should replace /*INSERT*/.

Special main method's name should be "main" (all characters in lower case), should be static, should have public access specifier and it accepts argument of String [] type (Varargs syntax String… can also be used). String [] argument can use any identifier name, even though in most of the cases you will see "args" is used. Position of static and public can be changed but return type 'void' must come just before the method name.

Let's check all the given options one by one:

public void static main(String [] args): Compilation error as return type 'void' must come just before the method name 'main'.

protected static void main(String [] args): Compiles successfully but as this method is not public, hence an Error regarding missing main method is thrown on execution.

public void main(String… args): Compiles successfully but as this method is not static, hence an Error regarding non-static main method is thrown on execution.

static public void Main(String [] args): Compiles successfully but as 'M' is capital in method 'Main', hence it is not special main method. An Error regarding missing main method is thrown on execution.

static public void main(String [] args): Valid definition, it compiles successfully and on execution prints "All is well" on to the console.

public static void main(String [] a): Valid definition, it compiles successfully and on execution prints "All is well" on to the console.

public static Void main(String [] args): Compilation error as Void is a final class in Java and in this case compiler expects main method to return a value of Void type. If you add `return null;` to the main method code will compile successfully but on execution an Error will be thrown mentioning that return type must be 'void' ('v' in lower-case).

public static void main(String… a): Valid definition, it compiles successfully and on execution prints “All is well” on to the console.

Correction Java Basics - 16

B - Only two options

Top-level class can use only two access modifiers [public and default(don’t specify anything)]. private and protected cannot be used.

As file name is M.java, hence class N cannot be public.

Top-level class can be final, hence it is a correct option.

Top-level class can be abstract and hence it is also a correct option.

Correction Java Basics - 17

`D – java –cp Quiz\Sec07\classes\ com.training.test.Exam`

To execute Exam class from WORK folder, you should specify the classpath (`Quiz\Sec07\classes\`) which contains whole path of the class(`com\training\test\Exam.class`).

And you should also use fully qualified name of the class, which is com.training.test.Exam.

Hence correct option is: `java –cp Quiz\Sec07\classes\ com.training.test.Exam`

Correction Java Basics - 18

D - ONE

Like any other method, main method can also be overloaded. But main method called by JVM is always with String [] parameter.

Don’t get confused with 10 as it is passed as “10”. Run above class with any command line arguments or 0 command line argument, output will always be ONE.

Correction Java Basics - 19

B -

```
public static void main(String [] a) {}
```

Special main method should have public access specifier and it takes argument of String [] type.

String [] argument can use any identifier name, even though in most of the cases you will see “args” is used.

Correction Java Basics - 20

D - 1 & 3 only

Planet is defined in ‘com.training.galaxy’ package, Creator is defined in ‘com.training.oca’ package and TestCreator is defined in ‘com.training.oca.test’ package.

Planet class doesn’t mention ‘Creator’ or ‘TestCreator’ and hence no import statements are needed in Planet class.

Creator class uses the name ‘Planet’ in its code and hence Creator class needs to import Planet class using ‘import com.training.galaxy.Planet;’ statement or ‘import com.training.galaxy.*;’ statement.

TestCreator class uses the name ‘Creator’ in its code and hence TestCreator class needs to import Creator class using ‘import com.training.oca.Creator;’ statement or ‘import com.training.oca.*;’ statement.

Please note, even though in TestCreator class, `Creator.create()` returns an instance of Planet class but as name ‘Planet’ is not used, hence Planet class is not needed to be imported.

Planet class correctly overrides toString() method, hence when an instance of Planet class is passed to println(…) method, as in the below statement:

System.out.println(Creator.create());

toString() method defined in the Planet class is invoked, which print “Planet: Earth” on to the console.

Correction Java Basics - 21

C - Two statements

Though given code looks strange but it is possible in java to provide same name to package, class (and constructor), variable and method.

Above code compiles successfully and on execution prints ONE on to the console. Constructor is not invoked as ‘new’ keyword is not used and that is why TWO will not be printed to the console.

In real world coding, you would not see such code and that is why it is a good question for the certification exam.

Correction Java Basics - 22

A - Welcome James Gosling!

Please note, if passed command line arguments contain space(s) in between, then it is a common practice to enclosed within double quotes. In this case “James Gosling” is passed as one String object and “Bill Joy” is also passed as one String object.

java Test “James Gosling” “Bill Joy” passes new String [] {“James Gosling”, “Bill Joy”} to args of main method. args[0] refers to “James Gosling” and args[1] refers to “Bill Joy”.

Hence, Welcome James Gosling! is printed on to the console. While printing the String object, enclosing quotes are not shown.

To use quotes as part of the String, you can escape those using backslash, such as:

java Test “"James Gosling”" “"Bill Joy”"

Above command will print Welcome “James Gosling”! on to the console.

Correction Java Basics - 23

D - Only two definitions

Special main method (called by JVM on execution) should be static and should have public access modifier. It also takes argument of String [] type (Varargs syntax String… can also be used).

String [] or String… argument can use any identifier name, even though in most of the cases you will see “args” is used.

final modifier can be used with this special main method.

Hence, from the given five definitions of main method, below two definitions will print expected output on to the console.

```
public static final void main(String... a) {
    System.out.println("Java Rocks!");
}
```

and

```
public static void main(String [] args) {
    System.out.println("Java Rocks!");
}
```

Correction Java Basics - 24

B - An exception is thrown at runtime

public static void main(String[] args) method is invoked by JVM.

Variable args is initialized and assigned with Program arguments. For example,

java Test: args refers to String [] of size 0.

java Test Hello: args refers to String [] of size 1 and 1st array element refers to “Hello”

java Test 1 2 3: args refers to String [] of size 3 and 1st array element refers to “1”, 2nd array element refers to “2” and 3rd array element refers to “3”.

Command used in this question: java Test Good, so args refers to String[] of size 1 and element at 0th index is “Good”.

args[1] = “Day!”; is trying to access 2nd array element at index 1, which is not available and hence an exception is thrown at runtime.

Correction Java Basics - 25

A - String

Like any other method, main method can also be overloaded. But main method called by JVM is always with String [] parameter. Don't get confused with 10 as it is passed as "10".

Execute above class with any command line arguments or 0 command line argument, output will always be "String".